BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

OPIC
OFFICE DE LA PROPRIÉTÉ INTELLECTUELLE DU CANADA

CIPO
CANADIAN INTELLECTUAL PROPERTY OFFICE

(12) (19) (CA) **Demande-Application**

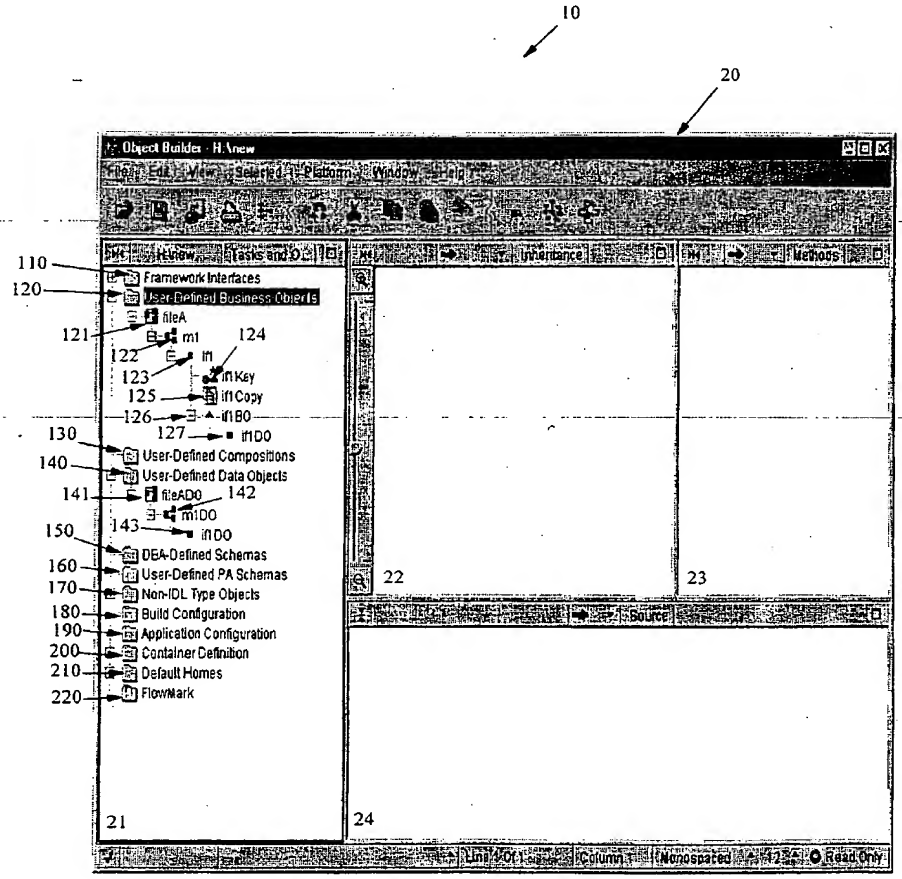
(21) (A1) **2,255,017**
(22) 1998/11/30
(43) 2000/05/30

(72) LAU, Christina P., CA
(71) IBM CANADA LTD. - IBM CANADA LIMITÉE, CA

(51) Int.Cl.[6] G06F 9/44, G06F 9/46, G06F 17/30, G06F 3/14

(54) **METHODE ET MECANISME RELATIFS A UN MODELE DE DONNEES XML ORIENTE TACHES**

(54) **METHOD AND MECHANISM FOR A TASK ORIENTED XML DATA MODEL**



(57) A task oriented data model for an object oriented development tool. The data model comprises a task oriented structure which mirrors the task tree embodied in the user interface of the object development tool. The object development tool exports the data model as a document expressed in meta data language such as XML. The XML data model document comprises a series of data elements which are arranged according to a meta data model to mirror the user interface task tree. The meta data model is implemented in XML as a Document Type Definition and comprises containment relationships defined by XML constructs. The task oriented data model provides granularity for exporting and importing data elements, and also facilitates locating data elements of interest in the data model.

TITLE: METHOD AND MECHANISM FOR A TASK ORIENTED XML DATA MODEL

REFERENCE TO RELATED APPLICATIONS

This application is related to the co-pending application filed in the name of the common assignee and entitled Method and Mechanism for a Task Oriented Data Model.

FIELD OF THE INVENTION

The present invention relates to object oriented programming systems, and more particularly to a task oriented structure for a data model.

BACKGROUND OF THE INVENTION

Traditionally in the computer programming arts, program code and data have been kept separate. For example, in the well-known C programming language, units of program code are called functions, and units of data are called structures. In C the functions and structures are not formally connected, that is, a function can operate on more than one type of data structure, and more than one function can operate on the same structure.

Object-oriented programming, in contrast, deals with objects, in which program code and data are merged into a single indivisible entity or "object". An object is an instance of an implementation

using the user interface in the object builder or by viewing the code generated from the binary file. Second, the data model must be imported or exported in its entirety, i.e. component parts of the data model cannot be imported or exported. This makes it much more time-consuming to perform impact analysis and to compare different model-generated outcomes. For example, to compare the modelling results obtained by modifying one object, it is usually necessary to import or export the entire data model, even though the data model is substantially unchanged in any respect other than the slightly altered object.

This problem has been addressed in other data models by using a proprietary language, for example, the known Rational Rose tool utilizes a Rose mdl format file to represent the meta-data for the data model. Because the mdl format has been well publicized, objects using the mdl format can be designed and analyzed by any modelling tool which supports mdl. Using a proprietary language such Rose mdl, the meta-data is organized so that the component parts of the data model, such as objects and task trees, can be imported and exported at various different levels of generality. However, using a proprietary language limits the inter-operability between the data model and other tools as these other tools will need to use parsers that are compatible with the proprietary language in order to retrieve meta-data from the data model.

Accordingly, there remains a need for generating a data model which is not confined to a proprietary language.

BRIEF SUMMARY OF THE INVENTION

The present invention provides a mechanism for a task oriented data model for a development tool in an object oriented programming system.

the capability to export and import data elements with varying degrees of granularity.

According to an aspect of the invention, there is provided a task oriented data structure embodied in a program storage device for an object oriented application program, the object oriented application program having a user interface comprising a plurality of items, said items defining a sequence of tasks for creating one or more objects, said data structure comprising: (a) a data model expressed in the form of a document according to a meta data language; (b) said data model including a plurality of data elements; (c) each of said data elements corresponding to one of the tasks in said sequence of tasks; and (d) said data elements being arranged according to a meta data model, wherein said meta data model mirrors the sequence of tasks defined in the user interface. Said meta data model may means for validating each of said data elements and the arrangement of said data elements. Further, said meta data language may comprise XML or Extensible Markup Language, and said data elements being arranged according to containment constructs specified in said meta data model. Said means for validating may also comprise a Document Type Definition specified in XML. And, said containment constructs may comprise tag definitions according to XML.

There is also provided, in an application program for creating objects, the application program having a user interface comprising a plurality of items, said items defining a sequence of tasks for creating the objects, and said application program including an export utility for exporting document files, a export utility comprising: (a) means for exporting a document file expressed in a meta data programming language; and (b) wherein said document file comprises a plurality of data elements, each of said data elements

validating may comprise a Document Type Definition specified in XML.

There is also provided a computer system for creating objects in an object oriented application program, the application program having a user interface comprising a plurality of items, said items defining a sequence of tasks for creating the objects, and said application program including an export utility for exporting document files, said computer system comprising: (a) means for outputting a document file, said document file being expressed according to a meta data programming language; and (b) said document file comprising a plurality of data elements, each of said data elements corresponding to one of the tasks in said sequence of tasks, and said data elements being arranged according to a meta data model, wherein said meta data model mirrors the sequence of tasks defined in the user interface. Said meta data model may include means for validating each of the data elements in said document file. Said meta data language may also comprise XML or Extensible Markup Language, and said data elements being arranged according to containment constructs specified in said meta data model. And, said means for validating may comprise a Document Type Definition specified in XML.

BRIEF DESCRIPTION OF THE DRAWINGS

Reference will now be made to the accompanying drawings which show, by way of example, preferred embodiments of the present invention, and in which:

Fig. 1 is a diagrammatic representation of the main window of an object builder of the type suitable for a task oriented data model according to the present invention; and

also the context for the content displayed in the three other panes 22, 23, and 24.

As shown in Fig. 1, the main view pane 21 comprises a "Framework Interfaces" item (or element) 110, a "User-Defined Business Objects" item 120, a "User-Defined Compositions" item 130, a "User-Defined Data Objects" item 140, a "DBA-Defined Schemas" item 150, a "User-Defined PA Schemas" item 160, a "Non-IDL Type Objects" item 170, a "Build Configuration" item 180, an "Application Configuration" item 190, a "Container Definition" item 200, a "Default Homes" item 210, and a "FlowMark" item 220.

The content of the other three panes 22, 23 and 24 in the window 10 depends on the selected item or element in the main view pane 21. Referring to Fig. 2, an item from "User-Defined Business Objects" item 120, specifically "Account", has been selected. In turn, the second pane 22 displays an inheritance graph for the Account item, and the third pane 23 shows the methods and attributes for the Account item. When one of the methods appearing in the third pane 23 is selected, the implementation of that method is shown in the fourth pane 24.

It will be appreciated that the object builder tool 10 utilizes a task tree structure. As shown in Fig. 1, the "User-Defined Business Objects" item 120 has a user interface which is organized as a task tree structure. The task-oriented structure of the user interface forces a user to follow a specified sequence in order to create a business object, i.e. an object containing business methods (logic) and data (state) that is intended for use within business applications. As will now be described, the data model according to the present invention comprises a task oriented structure which mirrors the user interface. The data model includes

user to define a module which comprises a grouping of interfaces. The third step performed by the user involves defining the public interface for the user-defined business object. The fourth step involves creating a key helper which comprises an object which provides a copy of the "identity" of the business object. The fifth step involves creating an object which duplicates everything about the business object, not just its identity. The sixth step comprises implementing the specific behaviour, i.e. methods, for the business object.

Similarly, according to the task tree structure for the "User-Defined Data Objects" item 140, a data object is created according to the following sequence of steps: (1) create file; (2) create module; and (3) create interface. A data object is an object that provides an object-oriented rendering of the application data.

According to the present invention, the data model comprises a task oriented structure which preserves the sequence of steps followed by a user during the creation of the object. The data content of the object, i.e. instance of data, is expressed by the object builder tool in the form of a text document. Since the data model mirrors the exact order in which the object was created by the user, the data model provides a structure from which the user interface of the object builder tool can be inferred. Because the user interface is implicit in the organization of the data model, the data model can be imported by another tool by simply following or scripting the data structure.

According to another aspect of the invention, the task oriented data model is implemented using the meta-data language XML (i.e. Extensible Markup Language) and a meta data model is implemented in the form of a Document Type Definition or DTD which

```
<!-- ************************** Module ***************************** -->
<!ELEMENT Module (#PCDATA | BOArtifact | DOArtifact | Comments | Constant |
Enumeration | Exception | Structure | Typedef | Union | Uuid)*>
<!ATTLIST Module Name CDATA #IMPLIED>

<!-- ***************************** DO File ************************** -->
<!ELEMENT DOFile (Comments | IncludeFile | Module | DOArtifact |
Constant | Enumeration | Exception | Structure | Typedef | Union | Uuid)*>
<!ATTLIST DOFile
  Name CDATA #IMPLIED>

<!-- ************************** BOArtifact ************************** -->
<!ELEMENT BOArtifact (Interface| KeyArtifact | CopyArtifact | BOImplementation)*>
<!ELEMENT Interface (#PCDATA | IsQueryable | IsWLM | Comments |
InterfaceInheritance | Attribute | Method | Constant | Enumeration |
Exception | Structure | Typedef | Union | Uuid |ObjectRelationship |
CompositeInfo)*>
<!ATTLIST Interface
  Name CDATA #IMPLIED>
<!ELEMENT IsQueryable (#PCDATA)*>
<!ELEMENT IsWLM (#PCDATA)*>
<!ELEMENT InterfaceInheritance (#PCDATA)*>

<!-- ************************** CompositeInfo ************************ -->
<!ELEMENT CompositeInfo (CompositeRules | InterfaceReference)*>
<!ATTLIST CompositeInfo
          CompositeToolMoniker CDATA #IMPLIED>

<!--************************** CompositeRules ************************ -->
<!ELEMENT CompositeRules (#PCDATA)*>

<!--************************* InterfaceReference ********************* -->
<!ELEMENT InterfaceReference EMPTY>
<!ATTLIST InterfaceReference
```

```
          InstanceKind CDATA #IMPLIED>

<!--****************************** DataStore ****************************** -->
<!ELEMENT DataStore (#PCDATA)*>
<!ATTLIST DataStore
          ReferencedCategory CDATA #IMPLIED
          ReferencedInterface CDATA #IMPLIED
          ReferencedName CDATA #IMPLIED
          ReferencedKind CDATA #IMPLIED
          InstanceInterface CDATA #IMPLIED
          InstanceName CDATA #IMPLIED
          InstanceKind CDATA #IMPLIED>

<!-- ****************************** Chain ****************************** -->
<!ELEMENT Chain EMPTY>

<!-- ****************************** Select ****************************** -->
<!ELEMENT Select (NullAttribute)*>

<!--****************************** NullAttribute ****************************** -->
<!ELEMENT NullAttribute (Prereq | DataStore | Chain | Select)*>
<!ATTLIST NullAttribute
          AttributeInterface CDATA #IMPLIED
          AttributeName CDATA #IMPLIED
          AttributeKind CDATA #IMPLIED
          Negated CDATA #IMPLIED
          ReferencedCategory CDATA #IMPLIED
          ReferencedInterface CDATA #IMPLIED
          ReferencedName CDATA #IMPLIED
          ReferencedKind CDATA #IMPLIED
          InstanceInterface CDATA #IMPLIED
          InstanceName CDATA #IMPLIED
          InstanceKind CDATA #IMPLIED>
```

```
<!--************************ LocationByKey ***************************-->
<!ELEMENT LocationByKey (InterfaceReference)*>
<!ATTLIST LocationByKey
          Group CDATA #IMPLIED
          FindObject CDATA #IMPLIED
          CreateObject CDATA #IMPLIED
          RemoveObject CDATA #IMPLIED
          CreateFromCopy CDATA #IMPLIED
          HomeName CDATA #IMPLIED
          FactoryFinderName CDATA #IMPLIED
          HomeParameter CDATA #IMPLIED>

<!--************************ Object Relationship ******************** -->
<!ELEMENT    ObjectRelationship    (Uuid    |    ObjectType    |
ReferenceCollectionImplementation | ReferenceResolvedByForeignKey |
HomeToQuery |Attribute | Method)*>
<!ATTLIST ObjectRelationship Name CDATA #IMPLIED>
<!ELEMENT ObjectType (#PCDATA)*>
<!ELEMENT ReferenceCollectionImplementation (#PCDATA)*>
<!ELEMENT ReferenceResolvedByForeignKey (#PCDATA)*>

<!-- ***************************** Construct ************************** -->
<!ELEMENT Constant (Type | Initializer | Size | Uuid)*>
<!ATTLIST Constant
  Name CDATA #IMPLIED>

<!ELEMENT Enumeration (Member | Uuid)*>
<!ATTLIST Enumeration
  Name CDATA #IMPLIED>

<!ELEMENT Exception (Member | Uuid)*>
<!ATTLIST Exception Name CDATA #IMPLIED>
```

```
<!ELEMENT StringBehaviour (#PCDATA)>
<!ELEMENT GetterImplementationType (#PCDATA)*>
<!ELEMENT SetterImplementationType (#PCDATA)*>
<!ELEMENT DOAttributeName (#PCDATA)*>
<!ELEMENT ParentInterface (#PCDATA)*>
<!ELEMENT UsedInImplementation (#PCDATA)*>
<!ELEMENT StaticData (#PCDATA)*>
<!ELEMENT GetterBodiesKeepAllPlatformsInSyncCpp (#PCDATA)*>
<!ELEMENT GetterBodiesKeepAllPlatformsInSyncJava (#PCDATA)*>
<!ELEMENT SetterBodiesKeepAllPlatformsInSyncCpp (#PCDATA)*>
<!ELEMENT SetterBodiesKeepAllPlatformsInSyncJava (#PCDATA)*>
<!ELEMENT GetterMethodBodyForPlatform (Language)*>
<!ATTLIST GetterMethodBodyForPlatform Name CDATA #IMPLIED>
<!ELEMENT SetterMethodBodyForPlatform (Language)*>
<!ATTLIST SetterMethodBodyForPlatform Name CDATA #IMPLIED>

<!--ELEMENT ImplementationLanguage ANY-->
<!ELEMENT Language (UseToolDef, UseFile, UseTDE,Filename?, ToolDef?,
UserDef?,((OOSQLSourceType?, OOSQLUserDef?) | (ESQLSourceType?, ESQLUserDef?))?)>
<!ATTLIST Language
  Name CDATA #IMPLIED>
<!ELEMENT UseToolDef (#PCDATA)>
<!ELEMENT UseFile (#PCDATA)>
<!ELEMENT UseTDE (#PCDATA)>
<!ELEMENT Filename (#PCDATA)*>
<!ELEMENT ToolDef (#PCDATA)*>
<!ELEMENT UserDef (#PCDATA)*>
<!ELEMENT OOSQLSourceType (#PCDATA)*>
<!ELEMENT OOSQLUserDef (#PCDATA)*>
<!ELEMENT ESQLSourceType (#PCDATA)*>
<!ELEMENT ESQLUserDef (#PCDATA)*>

<!--****************************** Method ***************************** -->
<!-- Preserve spaces and newline -->
<!ATTLIST Filename xml:space (default|preserve) "preserve">
```

```
<!ELEMENT ParameterType (#PCDATA)*>
<!ELEMENT DirectionalAttr (#PCDATA)*>
<!ELEMENT MethodType (#PCDATA)*>

<!--********************** DOArtifact ***************************** -->
<!ELEMENT DOArtifact (DOInterface| Comments)*>


<!--*************************  Key ********************************** -->
<!ELEMENT KeyArtifact (#PCDATA| File | Module | Interface | Attribute | Method
| InterfaceInheritance | Uuid | Prologue | CppPrologue | JavaPrologue | Epilogue
| CppEpilogue | JavaEpilogue | KeyAttrMapGroup)*>
<!ELEMENT File (#PCDATA | Uuid | IncludeFile | Comments )*>
<!ATTLIST File Name CDATA #IMPLIED>

<!--************************** Copy ********************************** -->
<!ELEMENT CopyArtifact (#PCDATA | File | Module | Interface | Attribute |
Method | InterfaceInheritance | Uuid | Prologue | CppPrologue |
JavaPrologue| Epilogue | CppEpilogue | JavaEpilogue)*>

<!--************************** BO Implementation ******************** -->
<!ELEMENT BOImplementation (#PCDATA| File | Module | Interface | Pattern |
LazyEvaluation | DataObjectInterface | SessionableBO | ImplementationLanguage |
UserData | ImplementationInheritance | Attribute | Method | Prologue |
CppPrologue | JavaPrologue | Epilogue | CppEpilogue | JavaEpilogue | Key | Copy
| Handle | DOInterface | MO | Uuid | LocationGroup | ObjectRelationship)*>
<!ELEMENT Pattern (#PCDATA)>
<!ELEMENT LazyEvaluation (#PCDATA)>
<!ELEMENT DataObjectInterface (#PCDATA)>
<!ELEMENT SessionableBO (#PCDATA)>
<!ELEMENT ImplementationLanguage (#PCDATA)*>
<!ELEMENT UserData (#PCDATA)*>
<!ELEMENT ImplementationInheritance (#PCDATA)>
<!ELEMENT Prologue (#PCDATA)*>
```

```
<!--************************** PO Stuff **************************** -->
<!ELEMENT PO (Uuid?, ShortName?, ModuleName?, JavaPackage?, JavaPath?,
PAOService?, POAccessType, Prologue?, CppPrologue?, JavaPrologue?, Epilogue?,
CppEpilogue?, JavaEpilogue?, BindFileName, Comments?, POAttribute*, POMethod*,
Schema?)>
<!ATTLIST PO
  Name CDATA #IMPLIED>
<!ELEMENT POAccessType (#PCDATA)>
<!ELEMENT BindFileName (#PCDATA)*>
<!ELEMENT POAttribute (Uuid?, Type, Size, ColumnName, KeySequenceNumber?, Key?,
Getter?,    Setter?,    GetterBodiesKeepAllPlatformsInSyncCpp?,
SetterBodiesKeepAllPlatformsInSyncCpp?,    GetterMethodBodyForPlatform*,
SetterMethodBodyForPlatform*)>
<!ATTLIST POAttribute Name CDATA #IMPLIED>
<!ELEMENT KeySequenceNum (#PCDATA)*>
<!ELEMENT KeySequenceNumber (#PCDATA)*>
<!ELEMENT POMethod (Uuid?, IsFrameworkMethod, MethodFunction?, Implementation,
ReturnType, POParameter*, PlatformSet?, MethodBodiesKeepAllPlatformsInSyncCpp?,
MethodBodyForPlatform*)>
<!ATTLIST POMethod Name CDATA #IMPLIED>
<!ELEMENT POParameter (Type, Size?, ParmAccessType?)>
<!ATTLIST POParameter Name CDATA #IMPLIED>
<!ELEMENT ParmAccessType (#PCDATA)>

<!--************************ DOImpl->PO Mappings ********************** -->
<!ELEMENT POMapping (AttributeMapping*, MethodMapping*)>
<!ELEMENT    AttributeMapping    (DataObjectAttribute,    PatternType?,
PrimitiveMapping?,((POAttributeImplementation+, MappingHelper?) | (KeyMapping |
StructMapping)*)?)>
<!ELEMENT DataObjectAttribute (#PCDATA)>
<!ELEMENT PatternType (#PCDATA)>
<!ELEMENT PrimitiveMapping (POAttributeImplementation+, MappingHelper?)>
<!ELEMENT POAttributeImplementation (POInstance, POInstanceAttribute)>
<!ELEMENT POInstance (#PCDATA)>
```

```
<!ELEMENT ContainedBySchemaGroup (#PCDATA)*>
<!ELEMENT DatabaseName (#PCDATA)>
<!ELEMENT UserName (#PCDATA)*>
<!ELEMENT SchemaName (#PCDATA)>
<!ELEMENT ShortName (#PCDATA)>
<!ELEMENT ModuleName (#PCDATA)>
<!ELEMENT SchemaType (#PCDATA)>
<!ELEMENT PKConstraint (#PCDATA)*>
<!ELEMENT PKConstraintComment (#PCDATA)*>
<!ELEMENT SQLFilename (#PCDATA)*>
<!ELEMENT JavaPackage (#PCDATA)>
<!ELEMENT JavaPath (#PCDATA)>
<!ELEMENT JavaClassName (#PCDATA)>
<!ELEMENT PAOService (#PCDATA)>
<!ELEMENT SchemaMethod (Uuid?, IsFrameworkMethod, MethodFunction?,
Implementation, ReturnType, SchemaParameter*)>
<!ATTLIST SchemaMethod Name CDATA #IMPLIED>
<!ELEMENT MethodFunction (#PCDATA)*>
<!ELEMENT SchemaParameter (Type, Size?)>
<!ATTLIST SchemaParameter Name CDATA #IMPLIED>
<!ELEMENT Column (ColumnName, ColumnType, NotNull, KeySequenceNumber,
ColumnSequenceNumber, Length, Scale, ForBitData, Comments)>
<!ATTLIST Column
  Name CDATA #IMPLIED>
<!ELEMENT ColumnName (#PCDATA)>
<!ELEMENT ColumnType (#PCDATA)>
<!ELEMENT NotNull (#PCDATA)>
<!ELEMENT ColumnSequenceNumber (#PCDATA)*>
<!ELEMENT Length (#PCDATA)>
<!ELEMENT Scale (#PCDATA)>
<!ELEMENT ForBitData (#PCDATA)>
<!ELEMENT Property (TypeString, TypeQualifier, Size, Key?, Getter?, Setter?,
Uuid?)>
<!ATTLIST Property
```

```
<!ELEMENT DatabaseType (#PCDATA)*>
<!ELEMENT DDLFilename (#PCDATA)*>
<!ELEMENT EditGeneratedFile (#PCDATA)*>

<!--****************************** MO Stuff ***************************-->
<!ELEMENT MO (File | Module | Interface | ImplementationInheritance |
              MOApplicationAdaptor | Uuid)*>
<!ELEMENT MOApplicationAdaptor (#PCDATA)*>
<!-- UDBO ? -->
```

As shown above the meta data model comprises a Document Type Definition (DTD) which specifies the set of required and optional elements, and their attributes, for the data models which are expressed as XML documents. In addition, the DTD specifies the names of the tags and the relationships among elements in the data model, i.e. XML document. It will also be appreciated that the task tree elements for the "User-Defined Business Objects" item 120 are expressed using containment relationships in the Document Type Definition shown above. For example, the file element or item 121 is defined as <!ELEMENT BOFile(Comments|....|Uuid)*>, and the module item 122 is defined as <!ELEMENT Module (#PCDATA|....Uuid)*>. Similarly, in the DTD the "attributes" of the object are defined as <!ELEMENT Attribute (Type |Initializer|....)*>, and the "methods" of the object are defined as <!ELEMENT Method (ReturnType|.....)*>. The specific syntax of the meta data model expressed in the above DTD will be within the understanding of those familiar with XML.

Next, the meta data model according to the present invention for the "Non-IDL Type Objects" item 170 is specified in the form a Document Type Definition as shown below.

```
<!ATTLIST MakefileOptionsForPlatformName CDATA #IMPLIED>
<!ELEMENT Uuid (#PCDATA)*>
<!ELEMENT Tier (#PCDATA)*>
<!ELEMENT ApplicablePlatforms (#PCDATA)*>
<!ELEMENT Description (#PCDATA)*>
<!ELEMENT LibraryName (#PCDATA)*>
<!ELEMENT MakeOptions (#PCDATA)*>
<!ELEMENT IDLCompile (#PCDATA)*>
<!ELEMENT JavaCompile (#PCDATA)*>
<!ELEMENT JCBJavaCompile (#PCDATA)*>
<!ELEMENT CPPCompile (#PCDATA)*>
<!ELEMENT LinkOptions (#PCDATA)*>
<!ELEMENT FilesInDLL (File)*>
<!ELEMENT File (#PCDATA)*>
<!ELEMENT LinkLibInDLL (LinkLib)*>
<!ELEMENT LinkLib (#PCDATA)*>
<!-- DLL ? -->
```

The specific syntax of the meta data model for the "Build Configuration" item as shown in the above DTD will be within the understanding of those familiar with XML.

The meta data model according to the present invention for the "Application Configuration" item 190 is specified in the form of a Document Type Definition as shown below.

```
<?xml encoding="US-ASCII"?>

<!ELEMENT ApplicationConfiguration (ApplicationFamilyUnit)*>
<!ELEMENT ApplicationFamilyUnit    (Description | Version | DiskSpace | Readme
| Application | Uuid)*>
<!ATTLIST ApplicationFamilyUnit
  Name CDATA #IMPLIED>
<!ELEMENT Description (#PCDATA)*>
```

```
<!ELEMENT State (#PCDATA)>
<!-- APPL ? -->
```

The specific syntax of the meta data model for the "Application Configuration" item as shown in the above DTD will be within the understanding of those familiar with XML.

The meta data model according to the present invention for the "Container Definition" item 200 is specified in the form of a Document Type Definition as shown below.

```
<?xml encoding="US-ASCII"?>

<!ELEMENT ContainerDefinition (Container)*>
<!ELEMENT Container (Uuid | Description | NumberOfComponents | TransactionPolicy
| TerminationPolicy | PassivationPolicy | PersistentReferences | BOPattern |
DOPattern | CachingServices | HomeServices | WorkLoadManaged | PolicyGroup |
UsesSession | SessionPolicy | SessionConnectionType | SessionConnectionName |
SessionPriority | PAAService )*>
<!ATTLIST Container  Name CDATA #IMPLIED>
<!ELEMENT Uuid (#PCDATA)*>
<!ELEMENT Description (#PCDATA)*>
<!ELEMENT NumberOfComponents (#PCDATA)*>
<!ELEMENT TransactionPolicy (#PCDATA)*>
<!ELEMENT TerminationPolicy (#PCDATA)*>
<!ELEMENT PassivationPolicy (#PCDATA)*>
<!ELEMENT PersistentReferences (#PCDATA)*>
<!ELEMENT BOPattern (#PCDATA)*>
<!ELEMENT DOPattern (#PCDATA)*>
<!ELEMENT CachingServices (#PCDATA)*>
<!ELEMENT HomeServices (#PCDATA)*>
<!ELEMENT WorkLoadManaged (#PCDATA)*>
<!ELEMENT PolicyGroup (#PCDATA)*>
<!ELEMENT UsesSession (#PCDATA)*>
```

Schemas" item 150 (Fig. 1); (6) all non-IDL types under the task tree for the "Non-IDL Type Objects" item 170 (Fig. 1); (7) all makefiles under the task tree for the "Build Configuration" item 180 (Fig. 1); (8) all application families under the task tree for the "Application Configuration" item 190 (Fig. 1); (9) all containers under the task tree for the "Container Definition" item 200 (Fig. 1).

Advantageously, the human-readable form of the data model (i.e. the XML document) and the meta data model (i.e. the DTD) allow a user to directly examine the data model and quickly locate the data element or item of interest.

The present invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. For example, while the task oriented data model was described in the context of an XML implementation, it will be understood that implementations using other meta data language specifications can be used. Therefore, the presently discussed embodiments are considered to be illustrative and not restrictive, the scope of the invention being indicated by the appended claims rather than the foregoing description, and all changes which come within the meaning and range of equivalency of the claims are therefore intended to be embraced therein.

The invention may be implemented as a computer program product comprising a program storage device and means recorded on said program storage device for instructing a computer to perform the steps of the invention and as a data structure embodied in a program storage device. Such a program storage device may include diskettes, optical discs, tapes, CD-ROMS, hard drives, memory

CLAIMS

The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. A task oriented data structure embodied in a program storage device for an object oriented application program, the object oriented application program having a user interface comprising a plurality of items, said items defining a sequence of tasks for creating one or more objects, said data structure comprising:

(a) a data model expressed in the form of a document according to a meta data language;

(b) said data model including a plurality of data elements;

(c) each of said data elements corresponding to one of the tasks in said sequence of tasks; and

(d) said data elements being arranged according to a meta data model, wherein said meta data model mirrors the sequence of tasks defined in the user interface.

2. The task oriented data structure embodied in a program storage device as claimed in claim 1, wherein said meta data model includes means for validating each of said data elements and the arrangement of said data elements.

3. The task oriented data structure embodied in a program storage device as claimed in claim 1 or claim 2, wherein said meta data language comprises XML or Extensible Markup Language, and said data

8. The export utility as claimed in claim 6 or claim 7, wherein said document file is expressed in XML or Extensible Markup Language, and said data elements being arranged according to containment constructs specified in said meta data file.

9. The export utility as claimed in claim 8, wherein said means for validating comprises a Document Type Definition specified in XML.

10. A computer program product for an object oriented application program for creating objects, the application program having a user interface comprising a plurality of items, said items defining a sequence of tasks for creating the objects, and said application program including an export utility for exporting document files, said computer program product comprising:

a program storage device;

means recorded on said program storage device for instructing a computer to perform the steps of,

(a) outputting a document file, said document file being expressed according to a meta data programming language; and

(b) said document file comprising a plurality of data elements, each of said data elements corresponding to one of the tasks in said sequence of tasks, and said data elements being arranged according to a meta data model, wherein said meta data model mirrors the sequence of tasks defined in the user interface.

11. The computer program product as claimed in claim 10, wherein said meta data model includes means for validating each of the data elements in said document file.

Language, and said data elements being arranged according to containment constructs specified in said meta data model.

17. The computer system as claimed in claim 16, wherein said means for validating comprises a Document Type Definition specified in XML.

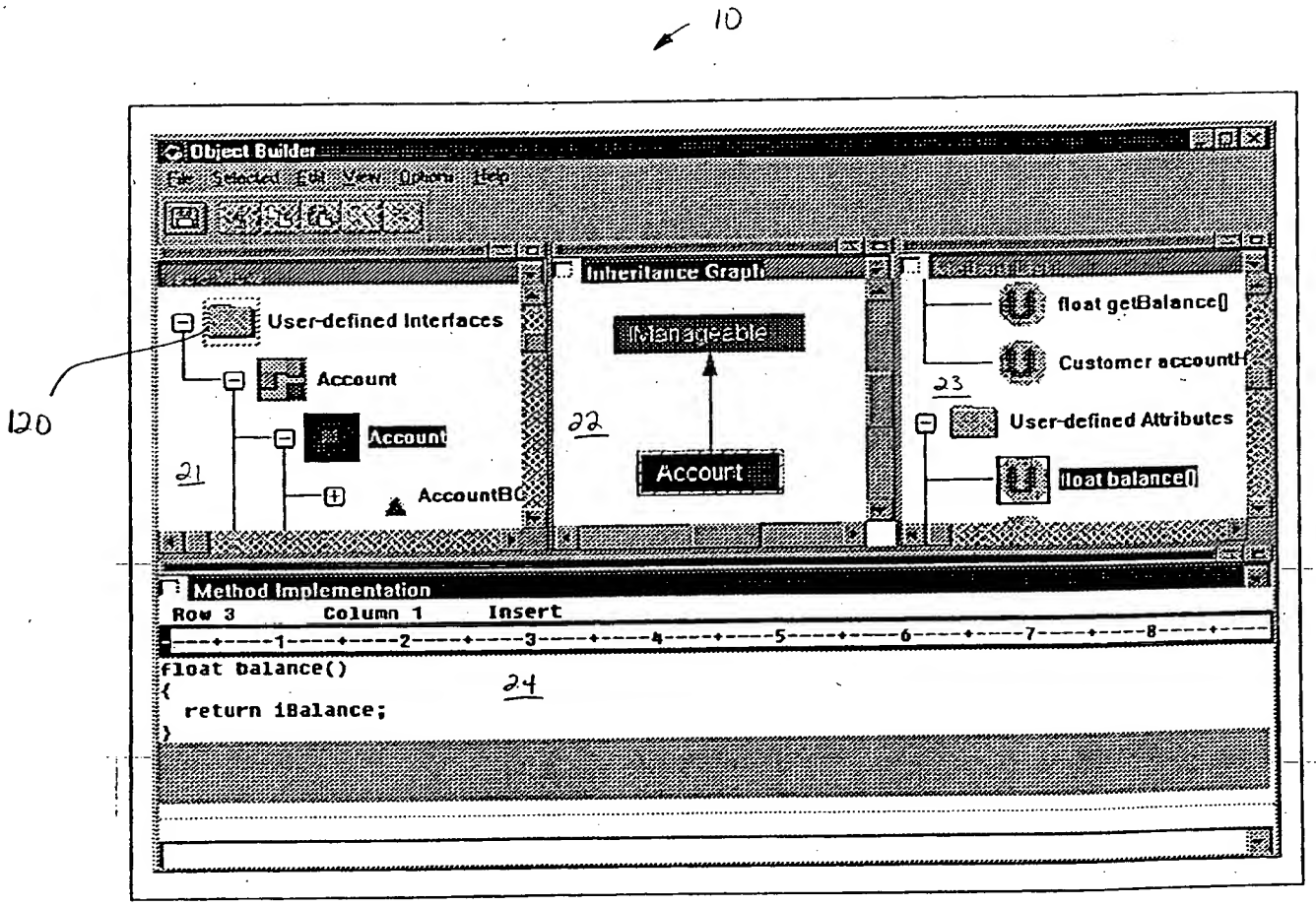

10
120
Object Builder
File Selected Edit View Options Help
User-defined Interfaces
Account
Account
AccountBC
21
Inheritance Graph
Manageable
22
Account
float getBalance()
Customer accountH
23
User-defined Attributes
float balance()
Method Implementation
Row 3 Column 1 Insert
float balance()
{
return iBalance;
}
24

FIG. 2